NAG Toolbox

nag_mesh_2d_gen_inc (d06aa)

1 Purpose

nag_mesh_2d_gen_inc (d06aa) generates a triangular mesh of a closed polygonal region in $\mathbb{R}^2$, given a mesh of its boundary. It uses a simple incremental method.

2 Syntax

```
[nv, nelt, coor, conn, ifail] = nag_mesh_2d_gen_inc(edge, coor, bspace, smooth, itrace, 'nvb', nvb, 'nvmax', nvmax, 'nedge', nedge, 'coef', coef, 'power', power)
[nv, nelt, coor, conn, ifail] = d06aa(edge, coor, bspace, smooth, itrace, 'nvb', nvb, 'nvmax', nvmax, 'nedge', nedge, 'coef', coef, 'power', power)
```

3 Description

nag_mesh_2d_gen_inc (d06aa) generates the set of interior vertices using a process based on a simple incremental method. A smoothing of the mesh is optionally available. For more details about the triangulation method, consult the D06 Chapter Introduction as well as George and Borouchaki (1998).

This function is derived from material in the MODULEF package from INRIA (Institut National de Recherche en Informatique et Automatique).

4 References

George P L and Borouchaki H (1998) *Delaunay Triangulation and Meshing: Application to Finite Elements* Editions HERMES, Paris

5 Parameters

5.1 Compulsory Input Parameters

1: **edge**$(\mathbf{3}, \mathbf{nedge})$ – INTEGER array

The specification of the boundary edges. **edge**$(1, j)$ and **edge**$(2, j)$ contain the vertex numbers of the two end points of the jth boundary edge. **edge**$(3, j)$ is a user-supplied tag for the jth boundary edge and is not used by nag_mesh_2d_gen_inc (d06aa).

Constraint: $1 \leq$ **edge**$(i, j) \leq$ **nvb** and **edge**$(1, j) \neq$ **edge**$(2, j)$, for $i = 1, 2$ and $j = 1, 2, \ldots,$ **nedge**.

2: **coor**$(\mathbf{2}, \mathbf{nvmax})$ – REAL (KIND=nag_wp) array

coor$(1, i)$ contains the x coordinate of the ith input boundary mesh vertex; while **coor**$(2, i)$ contains the corresponding y coordinate, for $i = 1, 2, \ldots,$ **nvb**.

3: **bspace**$(\mathbf{nvb})$ – REAL (KIND=nag_wp) array

The desired mesh spacing (triangle diameter, which is the length of the longer edge of the triangle) near the boundary vertices.

Constraint: **bspace**$(i) > 0.0$, for $i = 1, 2, \ldots,$ **nvb**.

4: **smooth** – LOGICAL

Indicates whether or not mesh smoothing should be performed.

If **smooth** = *true*, the smoothing is performed; otherwise no smoothing is performed.

5: **itrace** – INTEGER

The level of trace information required from nag_mesh_2d_gen_inc (d06aa).

itrace ≤ 0
No output is generated.

itrace ≥ 1
Output from the meshing solver is printed on the current advisory message unit (see nag_file_set_unit_advisory (x04ab)). This output contains details of the vertices and triangles generated by the process.

You are advised to set **itrace** = 0, unless you are experienced with finite element mesh generation.

5.2 Optional Input Parameters

1: **nvb** – INTEGER

Default: the dimension of the array **bspace**.

The number of vertices in the input boundary mesh.

Constraint: $3 \leq$ **nvb** $\leq$ **nvmax**.

2: **nvmax** – INTEGER

Default: the dimension of the array **coor**.

The maximum number of vertices in the mesh to be generated.

3: **nedge** – INTEGER

Default: the dimension of the array **edge**.

The number of boundary edges in the input mesh.

Constraint: **nedge** ≥ 1.

4: **coef** – REAL (KIND=nag_wp)

Default: 0.75.

The coefficient in the stopping criteria for the generation of interior vertices. This argument controls the triangle density and the number of triangles generated is in $O(\mathbf{coef}^2)$. The mesh will be finer if **coef** is greater than 0.7165 and 0.75 is a good value.

5: **power** – REAL (KIND=nag_wp)

Default: 0.25.

Controls the rate of change of the mesh size during the generation of interior vertices. The smaller the value of **power**, the faster the decrease in element size away from the boundary.

Constraint: $0.1 \leq$ **power** ≤ 10.0.

5.3 Output Parameters

1: **nv** – INTEGER

The total number of vertices in the output mesh (including both boundary and interior vertices). If **nvb** = **nvmax**, no interior vertices will be generated and **nv** = **nvb**.

2: **nelt** – INTEGER

The number of triangular elements in the mesh.

3: **coor**$(\mathbf{2}, \mathbf{nvmax})$ – REAL (KIND=nag_wp) array

coor$(1, i)$ will contain the x coordinate of the $(i - \mathbf{nvb})$th generated interior mesh vertex; while **coor**$(2, i)$ will contain the corresponding y coordinate, for $i = \mathbf{nvb} + 1, \ldots, \mathbf{nv}$. The remaining elements are unchanged.

4: **conn**$(\mathbf{3}, \mathbf{2} \times (\mathbf{nvmax} - \mathbf{1}))$ – INTEGER array

The connectivity of the mesh between triangles and vertices. For each triangle j, **conn**(i, j) gives the indices of its three vertices (in anticlockwise order), for $i = 1, 2, 3$ and $j = 1, 2, \ldots,$ **nelt**.

5: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, **nvb** < 3 or **nvb** $>$ **nvmax**,
or **nedge** < 1,
or **edge**$(i, j) < 1$ or **edge**$(i, j) >$ **nvb**, for some $i = 1, 2$ and $j = 1, 2, \ldots,$ **nedge**,
or **edge**$(1, j) =$ **edge**$(2, j)$, for some $j = 1, 2, \ldots,$ **nedge**,
or **bspace**$(i) \leq 0.0$, for some $i = 1, 2, \ldots,$ **nvb**,
or **power** < 0.1 or **power** > 10.0,
or $liwork < 16 \times$ **nvmax** $+ 2 \times$ **nedge** $+ \max(4 \times$ **nvmax** $+ 2,$ **nedge**$) - 14$,
or $lrwork <$ **nvmax**.

ifail $= 2$

An error has occurred during the generation of the interior mesh. Check the definition of the boundary (arguments **coor** and **edge**) as well as the orientation of the boundary (especially in the case of a multiple connected component boundary). Setting **itrace** > 0 may provide more details.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

Not applicable.

8 Further Comments

The position of the internal vertices is a function of the positions of the vertices on the given boundary. A fine mesh on the boundary results in a fine mesh in the interior. The algorithm allows you to obtain a denser interior mesh by varying **nvmax**, **bspace**, **coef** and **power**. But you are advised to manipulate the last two arguments with care.

You are advised to take care to set the boundary inputs properly, especially for a boundary with multiply connected components. The orientation of the interior boundaries should be in **clockwise** order and opposite to that of the exterior boundary. If the boundary has only one connected component, its orientation should be **anticlockwise**.

9 Example

In this example, a geometry with two holes (two interior circles inside an exterior one) is meshed using the simple incremental method (see the D06 Chapter Introduction). The exterior circle is centred at the origin with a radius 1.0, the first interior circle is centred at the point $(-0.5, 0.0)$ with a radius 0.49, and the second one is centred at the point $(-0.5, 0.65)$ with a radius 0.15. Note that the points $(-1.0, 0.0)$ and $(-0.5, 0.5)$) are points of 'near tangency' between the exterior circle and the first and second circles.

The boundary mesh has 100 vertices and 100 edges. Note that the particular mesh generated could be sensitive to the ***machine precision*** and therefore may differ from one implementation to another.

9.1 Program Text

```
function d06aa_example

fprintf('d06aa example results\n\n');

edge = zeros(3, 100, nag_int_name);
coor = zeros(2, 250);

% Define boundaries
ncirc     = 3; % 3 circles
nvertices = [40, 30, 30];
radii     = [1, 0.49, 0.15];
centres   = [ 0,   0;
             -0.5, 0;
             -0.5, 0.65];

% First circle is outer circle
csign = 1;
i1 = 0;
nvb = 0;
for icirc = 1:ncirc
  for i = 0:nvertices(icirc)-1
    i1 = i1+1;
    theta = 2*pi*i/nvertices(icirc);
    coor(1,i1) = radii(icirc)*cos(theta) + centres(icirc, 1);
    coor(2,i1) = csign*radii(icirc)*sin(theta) +  centres(icirc, 2);
    edge(1,i1) = i1;
    edge(2,i1) = i1 + 1;
    edge(3,i1) = 1;
  end
  edge(2,i1) = nvb + 1;
  nvb = nvb + nvertices(icirc);
  % Subsequent circles are inner circles
  csign = -1;
end
nedge = nvb;

% Initialise mesh control parameters
bspace = zeros(1, 100);
bspace(1:nvb) = 0.05;
smooth = true;
itrace = nag_int(0);

[nv, nelt, coor, conn, ifail] = d06aa( ...
        edge, coor, bspace, smooth, itrace);
fprintf('\nnv   = %d\n', nv);
```

```
fprintf('nelt = %d\n', nelt);
% Plot mesh
fig1 = figure;
triplot(transpose(double(conn(:,1:nelt))), coor(1,:), coor(2,:));
axis equal;
```

9.2 Program Results

```
     d06aa example results

nv   = 250
nelt = 402
```